ESTRUCTURAS DE DATOS

Curso 2018/19

PRÁCTICA 8

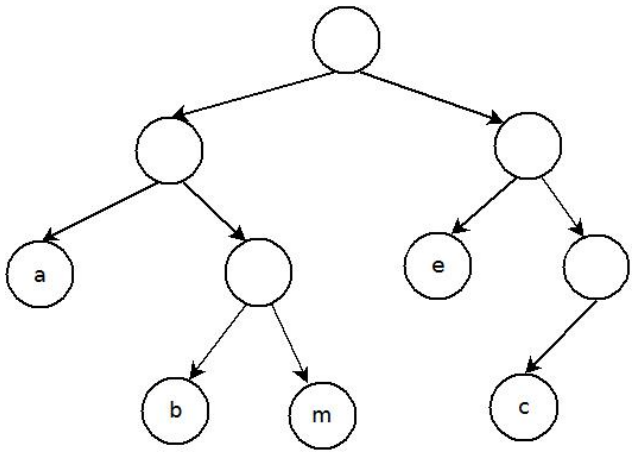
Árboles de Huffman

Instrucciones

- *Se debe completar en una sesión.*
- *Práctica individual.*
- *Lee el enunciado completo antes de comenzar. Los comentarios incluidos en el código también pueden proporcionar información útil.*
- *Se habilitará una tarea para que entregues el código desarrollado.*
- *La práctica será APTA si se superan todos los test de validación proporcionados.*

Los árboles de Huffman son árboles binarios que se utilizan para codificación de caracteres con fines de compresión/descompresión de ficheros. Un carácter se representa siempre con 8 o 16 bits. La idea básica es utilizar menos bits para representar aquellos caracteres más habituales en un fichero de texto.

Para ello el algoritmo de Huffman construye un árbol binario a partir de la colección de caracteres a codificar y sus frecuencias. El árbol de Huffman tiene los caracteres que codifica en los nodos hoja. El camino desde la raíz a un nodo hoja representa el código correspondiente al carácter en ese código hoja, asignando un bit 0 cada vez que toma una rama a la izquierda y un bit 1 cada vez que toma una rama a la derecha. Por ejemplo, dado el árbol de la figura, los códigos serían: a: 00; b:010; m:011; e:10; c:110.



La codificación de Huffman utiliza códigos de longitud variable, de modo que los caracteres con mayor frecuencia están en nodos a menor profundidad y por tanto, tienen códigos más cortos. El hecho de que los caracteres codificados estén sólo en nodos hoja hace que sean códigos unívocamente decodificables: ningún código es prefijo de otro código. Puedes encontrar más información aquí.

En esta práctica se construye un árbol de Huffman a partir de un fichero de texto, calculando la s frecuencias de aparición de cada uno de los caracteres. El fichero **`Frequences.java`** tiene una clase para guardar para cada uno de los 256 caracteres de la tabla ASCII su frecuencia de aparición de acuerdo con el texto del fichero de entrada. El fichero **`HuffmanTree.java`** contiene la clase que implementa los métodos propios para la codificación de Huffman.

La parte privada de la clase contiene las siguientes definiciones.

```
private static class Node {
      public boolean isLeaf = false; //Indica si el nodo es un nodo hoja
      public char c = '\0'; //Caracter almacenado
      public float f = (float) 0.0; //Frecuencia
      public Node left = null;
      public Node right = null;

      public Node(boolean leaf, char chr, float freq) {
            isLeaf = leaf; c = chr; f = freq; left = null; right = null;};
      }

      Node root = null; //El ábol binario de Huffman
```

El nodo tendrá un atributo para saber si es hoja o no, un caracter(por defecto '\0'), una frecuencia y referencias a los nodos hijo izquierdo y derecho. La clase **`HuffmanTree`** tendrá el atributo privado **`root`**(referencia al nodo raíz del árbol de Huffman.. El método que construye el árbol de Huffman a partir de las frecuencias se da ya implementado.

Ejercicio 1

Escribe el método privado **`List<Integer> findCode(char c)`** de la clase **`HuffmanTree.`** Obtiene la lista de unos y ceros que codifican el carácter **`c`** a partir del árbol de Huffman. Devuelve **`null`** si el carácter no es codificado por el árbol.

Ejercicio 2

Escribe el método privado **`void decode(String str, List<Character> l)`** de la clase **`HuffmanTree.`** A partir de una cadena de unos y ceros, **`str`**, que es la secuencia codificada de uno o más caracteres, devuelve la lista **`l`** de caracteres codificados.

Ejercicio 3

Escribe un constructor **`HuffmanTree(Map<Character, List<Integer>> codes)`** de la clase **`HuffmanTree.`** A partir de los caracteres codificados y la codificación de cada uno de ellos **`codes`**, construye el árbol de Huffman.